Projet de Programmation orienté objet

Jean-Rémy Falleri, Jonathan Narboni

October 2021

Introduction

Le but du projet est de développer une application de gestion de découpe de planche de bois. L'entreprise reçoit une ensemble de commandes de clients et doit découper les différentes planches demandées (de forme rectangulaire au départ) à partir de grands panneaux de bois achetés chez un fournisseur.

Le principe est donc d'optimiser de façon automatisée la découpe des planches de telle sorte à ce que l'entreprise perde le moins de bois (parce que la déforestation de l'amazonie c'est pas très 2021).

Le contexte

Le principe est donc de d'optimiser la découpe de rectangle de bois à partir de rectangles plus grands. Les commandes des clients et les disponibilités des fournisseurs sont spécifiées à partir de fichiers `fournisseur.xml`, et `client.xml`, et les découpes sont décrites par un fichier `decoupe.xml` ainsi qu'un fichier .svg `decoupe.svg` qui sera généra dans le même dossier que les fichiers des clients et des fournisseurs.

Les fournisseurs

Les fournisseurs peuvent fournir des panneaux de bois pour la découpe, la liste des informations donnée par les fournisseurs decrite dans le fichier `fournisseur.xml` est la suivante:

- La dimension des panneaux (longueur et largeur)
- Le prix unitaire des panneaux
- Le nombre de panneaux disponibles
- La date minimale de livraison des panneaux

Les clients

Les demandes des clients sont décrites dans le fichier `client.xml`, et sont les suivantes:

- La dimension des planches désirées
- Le nombre de planches désirées
- Le prix maximal possible
- La date maximale de livraison des panneaux

Les découpes

Le programme doit donc fournir aussi sous la forme d'un fichier .xml l'ensemble des panneaux à commander, ainsi que l'ensemble des découpes à effectuer

Etape 1

Le but de la première étape est de lire les fichier de commandes des clients, de disponibilités des fournisseurs, ainsi que des fichiers découpe et de vérifiier que ceux-ci sont bien valides.

Voici la liste des contraintes que les différents fichiers .xml devront vériifier:

- Pour les fournisseurs:
  - La longueur et la largeur des planches sont des entiers positifs
  - La longueur des planches est toujours supérieure à la largeur
  - La date limite de commandes est du type "JJ.MM.AA" où $\{J, M, A\} \subseteq \{1, \cdots, 9\}$ et est une date valide dans le futur.
  - Le prix unitaire est un nombre du type "aa.bb" strictement positif.
- Pour les clients:
  - La longueur et la largeur des planches sont des entiers positifs
  - La longueur des planches est toujours supérieur à la largeure
  - La date limite de commandes est du type "JJ.MM.AA" où $\{J, M, A\} \subseteq \{1, \cdots, 9\}$ et est une date valide dans le futur.
  - Le prix maximal de la commande est un nombre du type "aa.bb" strictement positif.
- Pour les découpes:
  - L'id du fournisseur, du panneau, du client et de la planche
  - Les positions des découpes sont des entiers positifs
  - Les dimensions des découpes sont des entiers positifs

- Les rectanglent ne se recouvrent pas
- Les découpes ne dépassent pas en dehors des panneaux
- L'entreprise ne peut pas perdre de l'argent sur les découpes
- Les dates fournisseurs et clients sont compatibles

Etape 2

Le but de cette étape est de donner un premier algorithme de découpe pour les planches, et de générer les fichiers .xml et .svg demandé. Le princpe de l'algorithme est basique : chaque découpe demande par le client sera effectuée sur un panneau different, les seules contraintes étant celle des dates, les clients étant prêts à payer une somme ridiculement grande pour leur commande. La génération du fichier .svg doit pouvoir être faite directement, ou à partir d'un fichier `découpe_1.xml` existant.

Etape 3

Le but de cette etape est d'implémenter deux autres algorithmes permettant d'obtenir de meilleurs résultats. Le programme devra pouvoir fournir les fichiers .xml et .svg pour chacun des algorithme différents, ainsi qu'une manière de comparer les résultats des différents algorithmes.

1. Le premier algorithme est très simple:

   - Les planches (P_i) à découper sont ordonnées lexicographiquement dans l'ordre décroissant suivant (L,l), où L est la longueur de la planche et l sa largeur.
   - Chaque planche est positionné à l'abscisse 0, et tant qu'on peut placer des planches supplémentaires, on le fait
   - On prend une nouvelle planche quand cela n'est plus possible.

2. Le deuxième algorithme est plus optimisé:

   - Les planches (P_i) à découper sont ordonnées lexicographiquement dans l'ordre décroissant suivant (l,L), où L est la longuueur et l la largeur de la planche à découper.
   - La première planche P_0 est placée en (0,0)
   - Puis tant que c'est possible on place successivement le maximum de planches P_i à gauche de la première planche P_0, avec $L(P_i) \leq L(P_0)$
   - Quand ce n'est plus possible, on place la planche restante avec la plus grande largeur en dessous de P_0, et on réitère le processus.

Etape 4

L'étape finale est aussi d'implémenter un algorithme permettant de prendre aussi en commande des clients n'importe quel polygone. Un polygone etant décrit comme une sequence de points dans le plan d'aire strictement positive. Le programme doit être capable d'effectuer toutes les étapes précédentes en tenant compte de la nouvelle forme possible des planches.

Évaluation

Ce projet est à réaliser par groupe de 2 (3 si besoin) étudiants. Vous serez évalués sur :

- La conception des classes et l'architecture de votre programme.
- La modularité et l'extensibilité du programme (minimiser l'effort pour inclure de nouveaux type de contraintes, de commande etc).
- L'utilisation des concepts fondamentaux de la programmation orientée objet.
- La robustesse de la façon dont vous avez implémenté les fonctionnalités.
- La gestion des erreurs.

Fonctionnalités supplémentaires

Des points bonus pourront être obtenus pour chaque implémentation d'une fonctionnalité proposée dans cette liste:

- Implémenter des contraintes supplémentaires pour les clients/fournisseur : dureté du bois, origine, couleur, épaisseur
- Une interface graphique permettant d'afficher une fenetre avec les découpes effectuées à la fin du programme
- Implémenter la possibilité des faire des découpes 3D, et de donner les résultats sous la forme uniquement sous la forme d'un fichier `.xml`
- Interfacer de votre programme avec un solveur de programme lineaire pour obtenir un algorithme donnant une réponse optimale

N'oubliez pas d'expliquer dans le fichier README ce que vous avez implémenté.

Contraintes sur le code

Vous devez absolument respecter ces contraintes. Une pénalité de 2 points sera appliquées sur chaque projet qui ne respecte pas:

- Votre projet contient plusieurs paquetages.
- Les noms des paquetages sont intégralement en minuscules.
- Les noms des classes et interfaces commencent par une majuscule.
- Les noms des méthodes, des attributs et des variables commencent par une minuscule.
- Utilisez le CamelCase pour les noms des classes, interfaces, méthodes et attributs.
- Choisissez une langue pour tout votre projet (de préférence l'anglais), et gardez ce même standard pour tout le projet.
- Les attributs ont une visibilité *private* ou *protected*. Des accesseurs permettent si nécessaire d'accéder en lecture et/ou écriture aux attributs.
- La visibilité des méthodes dépend de leur utilisation.
- Les constructeurs sont définis uniquement s'ils sont nécessaires.
- Le code des méthodes compliquées doit être commenté.

Le programme doit pouvoir être lancé grace à la ligne de commande:

```
java decoupe nom_du_dossier
```

où `decoupe` est le nom du programme est nom_du_dossier est le nom du dossier contenant les fichiers `fournisseur.xml` et `client.xml` au début, et contenant le fichier `decoupe_X.xml` et `decoupe_X.svg` correspondants au plan de découpe de l'algorithme numero `X` après avoir lancé le programme.

Livrable

Il sera demandé aux étudiants de fournir une archive contenant le code source du programme. À la racine de l'archive, il doit y avoir un fichier README spécifiant les étapes qui ont été complétées et celles qui ne l'ont pas été, ainsi que les fonctionnalités supplémentaires qui ont été implémentées. L'archive doit aussi contenir un fichier java archive decoupe.jar qui permet de lancer le jeu à l'aide de la commande `java -jar decoupe.jar nom_du_dossier`. Le schéma de nommage de l'archive doit être le suivant (pénalité de 2 points en cas de non respect) : pg220-2020-LOGINS.zip, où LOGINS est la liste des logins des membres du groupe, classée par ordre alphabétique et séparés par le symbole

"_". L'archive est à déposer, avant le **XX/12/2021** à 23h59, via la plateforme http://moodle.ipb.fr.

L'archive devra contenir au moins une version du programme correspondant à la version de base du sujet (les quatre étapes) avec un fichier ".jar" exécutable (voir le readme des tests pour en créer un) et un dossier contenant les sources. Si vous avez aussi effectué une version avec des fonctionnalités supplémentaires, elle devra être dans un dossier séparé (".jar" + sources correspondantes).

Les tests

Les tests seront à effectuer à partir des fichiers `.xml` correspondants à chaque étapes (chaque ensemble de fichier correspondant à l'aetape i se trouvera dans le dossier etape i.